ISSN 2413–9599 (Print)
ISSN 2541–870X (Online)

www.radioprom.org

DOI: 10.21778/2413-9599-2019-29-3-55-67
УДК 004.052.2

Подход к проектированию динамически реконфигурируемых блоков арбитража для встраиваемых систем

Е. А. Суворова[1]

[1] Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия

В настоящее время активно развивается направление разработки динамически реконфигурируемых компонентов для встраиваемых систем на базе FPGA. Однако проекты, разработанные на FPGA, очень сильно уступают по своим характеристикам проектам, реализованным по технологии ASIC с использованием тех же проектных норм. Это существенно ограничивает область применения реконфигурируемых систем на базе FPGA. В статье показана актуальность динамической реконфигурации блоков арбитража, предназначенных для встраиваемых систем. Рассмотрены существующие методы проектирования динамически реконфигурируемых компонентов для технологии ASIC и выполнена оценка их применимости для разработки блоков арбитража – сложных функциональных блоков систем-на-кристалле и сетей-на-кристалле. Предложен подход к проектированию динамически реконфигурируемых блоков арбитража для встраиваемых систем, позволяющий учесть специфические требования к этим блокам.

Ключевые слова: динамическая реконфигурация, алгоритмы арбитража, системы-на-кристалле, сети-на-кристалле, встраиваемые системы

Для цитирования:
Суворова Е. А. Подход к проектированию динамически реконфигурируемых блоков арбитража для встраиваемых систем // Радиопромышленность. 2019. Т. 29, № 3. С. 55–67. DOI: 10.21778/2413-9599-2019-29-3-55-67

The approach to design of dynamically reconfigurable arbitration units in embedded systems

E. A. Suvorova[1]

[1] Saint-Petersburg State University of Aerospace Instrumentation, Saint-Petersburg, Russia

Today we are seeing an intensive development of dynamically reconfigurable components in the FPGA-based embedded systems. Nevertheless, by their parameters, FPGA-based projects are essentially inferior to those that are on ASIC and the same design rules. This significantly limits applications of the FPGA-based reconfigurable systems. The paper presents relevance of dynamic reconfiguration for arbitration units in embedded systems. There is a review of existing design techniques for ASIC-based dynamically reconfigurable components. They have been also evaluated by applicability for the arbitration unit development (complex function modules for systems-on-chip and networks-on-chip). The authors have proposed the approach to the development of dynamically reconfigurable arbitration units in embedded systems. The approach makes it possible to consider specific requirements to these units.

Keywords: dynamic reconfiguration, arbitration algorithms, system-on-chip, network-on-chip, embedded systems

For citation:
Suvorova E. A. The approach to design of dynamically reconfigurable arbitration units in embedded systems. Radio industry (Russia), 2019, vol. 29, no. 3, pp. 55–67. (In Russian). DOI: 10.21778/2413-9599-2019-29-3-55-67

Введение

Для большинства современных локальных сетей и сетей-на-кристалле для встраиваемых систем одним из необходимых свойств является возможность изменения режима работы в ходе функционирования. Переход в другой режим функционирования может осуществляться при изменении набора решаемых в системе задач или вследствие возникновения ошибок в компонентах [1–5].

В разных режимах функционирования могут различаться пути передачи данных в сети, добавляться новые потоки, отключаться какие-то из существующих потоков. Также изменяются характеристики передаваемых потоков, появляются новые требования/правила для обработки потоков данных – качества сервиса, которые должны обеспечиваться при передаче данных через сеть [2, 3, 5]. Это приводит к необходимости изменения схем и правил арбитража. Таким образом, адаптация блоков арбитража к различным режимам работы, в частности к новым режимам (характеристики которых были определены уже после разработки микросхемы), является очень важной характеристикой, которая позволяет существенно продлить срок службы оборудования.

Потенциально блок арбитража может быть реализован полностью аппаратно, полностью программно или аппаратно-программно. Полностью программная реализация предоставляет очень широкие возможности по динамической реконфигурации за счет возможности изменения программного обеспечения. Однако в большинстве встраиваемых систем выделенное процессорное ядро не может использоваться для выполнения функциональности блока арбитража из-за слишком больших накладных расходов по площади и энергопотреблению. При использовании аппаратной реализации динамическая реконфигурация может достигаться за счет использования технологии FPGA. В этом случае смена прошивки позволяет полностью изменить логику работы блока арбитража.

Требования к блоку арбитража, его типовая структура и архитектура

Блок арбитража получает запросы на использование некоторого ресурса, по некоторому правилу выбирает один из них и выдает грант на использование ресурса. Например, в качестве ресурса может выступать выходной порт маршрутизатора пакетов. Блок арбитража получает запросы на передачу пакетов в выходной порт от входных портов маршрутизатора и выбирает, из какого входного порта будет передаваться следующий пакет в данный выходной порт. Также в качестве ресурса может выступать, например, точка подключения ведомого устройства к коммутатору или шине AXI, AHB, физический канал передачи данных, по которому могут передаваться пакеты, фреймы, флиты (flit (flow control unit) – атомарная (неделимая) часть передаваемого потока данных) данных из разных виртуальных каналов.

Арбитраж должен выполняться за короткое время. Для многих встраиваемых систем приемле-

мое значение – 2–10 тактов локальной частоты. Правило, по которому осуществляется выбор запроса, зависит от режима функционирования.

В разных режимах функционирования системы может требоваться поддержка различных качеств сервиса (уровней приоритета, выделенной гарантированной пропускной способности, планирования) для разных потоков данных. Для поддержки различных качеств сервиса требуется использовать разные правила арбитража. Приоритеты (и другие параметры качества сервиса) могут быть заданы для каждого источника запроса (однозначно определяться номером источника запроса) или в привязке к передаваемым объектам данных. Например, уровень приоритета пакета при передаче в выходной порт может задаваться в привязке к заголовку пакета. При этом порядковый номер входного порта, из которого поступил пакет, может не учитываться при арбитраже.

Был выполнен анализ алгоритмов работы блоков арбитража, используемых в современных системах и сетях-на-кристалле для встраиваемых систем [22–28]. Необходимо отметить, что в рамках современных стандартов внутрикристальных коммуникаций, ориентированных на использование шин и коммутаторов, таких как AHB, AXI, WISHBONE и др., не специфицированы алгоритмы арбитража. Соответственно, одни и те же алгоритмы арбитража могут использоваться для различных стандартов. Аналогична ситуация и для многих сетей-на-кристалле. В них могут использоваться различные алгоритмы арбитража, в том числе и такие же, как в системах на базе шин и коммутаторов.

Входными данными для работы алгоритмов арбитража является вектор запросов ресурса. Также могут использоваться параметры источников запросов (например, уровни приоритетов, выделенная доля пропускной способности и др.), информация о моментах времени передачи очередного слова (например, для контроля фактически использованной пропускной способности), номер текущего таймслота и др. Выходными данными является вектор грантов на предоставление ресурса. Как можно видеть, набор входных и выходных данных не зависит от используемого в системе стандарта коммуникаций.

В разных режимах функционирования системы могут использоваться статические или динамические приоритеты, статически задаваемые уровни приоритетов для разных источников могут быть разными в разных режимах. Если в одном из режимов работы несколько источников имеют одинаковый статический уровень приоритета, может потребоваться циклическая динамическая смена приоритетов.

В разных режимах функционирования системы могут использоваться разные схемы обеспечения гарантированной пропускной способности. Например, в одном из режимов может использоваться метод «дырявого ведра», позволяющий блокировать слишком частую отправку объектов данных каждым из источников. В другом режиме выделенная доля пропускной способности может обеспечиваться за счет контроля соотношения количества фактически переданных объектов данных и выделенной доли пропускной способности для каждого из источников.

Если в некоторых режимах функционирования используется планирование, то в системе существует расписание, в рамках которого для каждого источника передача данных разрешена в рамках одного или нескольких таймслотов.

В разных режимах функционирования передача данных может быть прерываемая или непрерываемая. Если используется прерываемая схема передачи данных, при поступлении запроса с более высоким приоритетом текущая передача данных, соответствующая запросу с более низким приоритетом, может быть временно прервана. Также могут поддерживаться режимы, в которых передача данных прерывается по истечении таймслота, в котором разрешена передача данных этого типа, после передачи некоторого количества символов данных или по другим условиям. Соответственно, может возникать необходимость контролировать выполнение/невыполнение нескольких условий одновременно – параллельно обрабатывать несколько потоков событий.

Таким образом, реконфигурируемый блок арбитража должен удовлетворять следующим требованиям и ограничениям:

- поддержка правил арбитража, которые заранее известны;
- поддержка правил арбитража, которые заранее не известны;
- ограничения по площади и энергопотреблению;
- требования по быстродействию (в том числе возможность параллельной обработки нескольких потоков событий).

В соответствии с требованиями к блоку арбитража, а также с учетом того, что при реализации встраиваемых систем (на этапе логического и физического синтеза) должны использоваться стандартные системы автоматизированного проектирования (САПР) и технологические процессы, мы сформулировали набор критериев для выбора методов динамической реконфигурации:

- накладные расходы по площади и энергопотреблению должны быть ограничены (в соответствии с требованиями пользователя);

Рисунок 1. Обобщенная схема автомата блока арбитража
Figure 1. Generalized chart of arbitration unit state-machine

- должна существовать возможность обработки потока данных в реальном времени при использовании рабочей частоты, соответствующей скорости поступления данных (работа на более высокой частоте приводит к существенному увеличению энергопотребления, что недопустимо для многих применений);
- должна существовать возможность параллельной обработки нескольких потоков событий в реальном времени при использовании рабочей частоты, соответствующей скорости поступления данных;
- должна существовать возможность реализации новых режимов;
- не должно быть специфических требований к технологии реализации;
- должна существовать возможность использования стандартных САПР.

Обобщенная схема автомата, соответствующего блоку арбитража, представлена на рис. 1.

Сплошными линиями обозначены состояния и переходы, которые имеются в любом автомате арбитража. Пунктирными линиями обозначены примеры дополнительных состояний и переходов, которые могут быть нужны для реализации некоторых режимов.

Обзор методов обеспечения динамической реконфигурации при использовании технологии ASIC

При использовании технологии ASIC динамическая реконфигурация может быть обеспечена на технологическом (или близком к технологическому) уровне либо на более высоких уровнях представления, которые могут быть реализованы за счет использования методов, относящихся к более низким уровням представления.

На технологическом уровне динамическая реконфигурация может достигаться за счет использования:

- библиотек логических элементов, допускающих возможность конфигурирования [6, 7];
- eFPGA [8–10].

Применимость библиотек логических элементов, допускающих возможность конфигурирования, сильно ограничена большим количеством памяти, необходимым для хранения конфигурации, и отсутствием поддержки в САПР. Существующие блоки eFPGA доступны для небольшого количества производственных технологий. Такие блоки имеют объем, на порядки больший, чем необходимо для реализации блока арбитража. Поэтому они плохо подходят для его реализации.

На близком к технологическому уровне динамическая реконфигурация может достигаться за счет использования:

- таблиц соответствия, реализованных на библиотечных компонентах, которые доступны для ASIC (блочной памяти или триггерах);
- структур, аналогичных программируемых логических матриц (ПЛМ), реализованных на библиотечных компонентах (например, AND, OR, NOT).

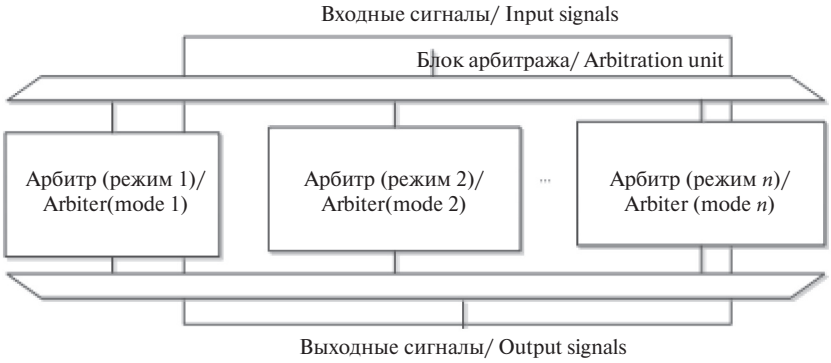

Рисунок 2. Реконфигурируемый блок арбитража на базе схемы, обеспечивающей включение/отключение отдельных элементов
Figure 2. Reconfigurable arbitration unit based on the scheme that enables individual element switching on/off

Такого рода структуры могут быть описаны на языках описания аппаратуры высокого уровня как soft-блоки и синтезированы стандартными САПР для ASIC. Их использование ограничивается ощутимыми накладными расходами по площади и ограничениями по достижимому быстродействию. Эти блоки могут использоваться как технологическая основа для реализации реконфигурируемых блоков на более высоких уровнях представления, где могут использоваться:

- схемы, обеспечивающие включение/отключение отдельных элементов (использование избыточности на уровне компонентов и связей);
- реконфигурируемые автоматы;
- реконфигурируемые DataPath;
- ядра микропроцессоров;
- ядра специализированных микроконтроллеров.

Схемы, обеспечивающие включение/отключение отдельных элементов, логически строятся на базе мультиплексоров. При их синтезе с использованием стандартных САПР такие конструкции реализуются на базе стандартных библиотечных компонентов. Они имеют ощутимо большую площадь и меньшее быстродействие по сравнению со специализированными ключами, используемыми для аналогичных целей в FPGA, что необходимо учитывать при проектировании. Применительно к блоку распределения пакетов в простейшем случае такая схема может выглядеть так, как показано на рис. 2.

К достоинствам данного подхода можно отнести простоту реализации, но площадь блока, спроектированного таким образом, может оказаться неприемлемо большой, если требуется поддерживать большое количество режимов. Другой существенный недостаток этой схемы – невозможность конфигурирования для новых режимов.

Другой способ обеспечения динамической реконфигурации – использование динамически реконфигурируемых автоматов [11–13]. При их использовании могут изменяться правила перехода между состояниями, логический смысл состояний, значения выходных сигналов, с ними связанных, количество состояний. Автоматы обеспечивают быструю (в течение одного такта работы устройства) реакцию на возникающие события, что является очень существенным достоинством при реализации блоков арбитража. Динамически реконфигурируемый автомат может иметь разные варианты физической реализации. В простейшем случае он может быть реализован на базе таблицы соответствия, на входы которой подаются значения входных сигналов автомата и его текущее состояние, а в строках таблицы соответствия записаны значения выходных сигналов и следующее состояние. Однако из-за больших накладных расходов по площади такой вариант реализации подходит только для автоматов с малым количеством состояний и входных сигналов, так как количество ячеек памяти в таблице соответствия определяется по формуле $2(N_i + N_s)$, где N_i – суммарная разрядность вектора входов; N_s – разрядность вектора состояния. Для сокращения накладных расходов по площади могут использоваться:

- декомпозиция исходного автомата на подавтоматы [12, 13, 16];
- дополнительные схемы мультиплексирования входных сигналов, позволяющие сократить количество сигналов, подаваемых на вход таблицы соответствия (N_i) по сравнению с исходным количеством входов автомата [14, 15].

Также для реализации динамически реконфигурируемых автоматов могут использоваться структуры на базе ПЛМ, дополненные регистром для хра-

нения текущего состояния автомата. Основным ограничением при использовании этого способа реализации является размер памяти, необходимой для хранения текущей конфигурации ПЛМ.

Поскольку решение о переходе из состояния в состояние для динамически реконфигурируемого автомата принимается каждый такт и может приниматься с учетом нескольких различных условий, динамически реконфигурируемые автоматы хорошо подходят для обработки нескольких потоков событий в реальном времени.

Динамически реконфигурируемые DataPath представляют собой структуры, состоящие из функциональных модулей и системы связей между ними. Динамически реконфигурируемыми могут быть и сами функциональные модули, и система связей между ними. Как правило, реконфигурируемые DataPath применяются для непосредственной обработки данных, а логика управления реализуется другими способами, например с использованием динамически реконфигурируемых автоматов [17]. Достоинство реконфигурируемых DataPath при реализации блоков арбитража – возможность параллельного контроля нескольких потоков событий. Например, DataPath может включать в себя несколько счетчиков, использующихся для подсчета количества событий разных типов.

Еще один способ обеспечения динамической реконфигурации – использование ядер микропроцессоров и микроконтроллеров [18–21]. Однако для реализации алгоритмов арбитража в реальном времени требуется поддержка нескольких потоков событий (отслеживание состояний линий запросов, состояний различных таймеров и счетчиков). Это приводит к тому, что ядро должно функционировать на частоте, которая на порядок и более превосходит скорость поступления данных. В результате накладные расходы на использование ядер процессоров и микроконтроллеров оказываются недопустимо большими для реализации блоков арбитража.

В таблице приведено сравнение рассмотренных методов по определенным нами в предыдущем разделе критериям.

Из таблицы видно, что наиболее подходящими для проектирования динамически реконфигурируемых блоков арбитража с учетом допустимых аппаратных затрат, требуемых временных характеристик и возможности обработки нескольких потоков событий являются системы на базе реконфигурируемых автоматов и реконфигурируемых DataPath.

Предлагаемый подход для разработки динамически реконфигурируемого блока арбитража

В рамках данного подхода реконфигурируемый автомат предлагается использовать для реализации машины состояний, соответствующей алгоритму арбитража. При этом действия по анализу заголовка пакета (флита), выбору источника запроса, которому будет предоставлен ресурс, формированию выходных сигналов блока арбитража и входных сигналов для реконфигурируемого автомата предлагается выполнять в реконфигурируемом DataPath. Каждому состоянию реконфигурируемого автомата соответствует вектор конфигурации, подаваемый на DataPath. Если сравнить предлагаемую систему с процессорным ядром, можно провести параллель между ролью векторов конфигурации и ролью команд процессора во VLIW-архитектуре.

Также можно провести параллель между действиями блока управления и условными и безусловными переходами, выполняемыми процессорным ядром. При этом текущие значения условий переходов формируются в DataPath. DataPath включает в себя несколько параллельно функционирующих функциональных модулей (ФМ). Это позволяет одновременно (в течение одного такта системной частоты) выполнять несколько действий над одним (очередным) словом заголовка пакета (флита). Параллельно (в этом же такте) с этим могут выполняться прочие действия, например подсчет тайм-аутов, количества обработанных слов и другие, определение текущих значений условий перехода в следующее состояние. Таким образом, при частоте работы, соответствующей скорости поступления данных, обеспечивается обработка потока данных и других потоков событий.

В рамках предлагаемого подхода реконфигурируемый блок арбитража (рис. 3) имеет следующую структуру: блок управления на базе реконфигурируемого автомата, реконфигурируемый DataPath, подсистему памяти и блок внутреннего интерфейса с FPGA-подобной структурой (обеспечивает интерфейс между регистрами и DataPath, а также между ФМ DataPath). Блок арбитража имеет функциональный интерфейс, который включает в себя сигналы запроса ресурса, сигналы гранта на предоставление ресурса, сигналы для передачи значений дополнительных параметров, характеризующих обмен данными, и конфигурационный интерфейс, через который в блок записывается очередная конфигурация.

Конфигурация включает в себя набор векторов конфигурации (записываются в память конфигурационных векторов), начальные значения регистров и значения, записываемые в таблицу соответствия, являющуюся основой реконфигурируемого автомата. Каждый вектор конфигурации состоит из подвекторов, представляющих собой значения, которые подаваются на управляющие входы блоков, входящих в состав DataPath, и блок внутрен-

Таблица. Результаты сравнения методов обеспечения динамической реконфигурации
Table. Results of comparison between techniques of dynamic reconfiguration

Критерий/ метод динамической реконфигурации/ Dynamic reconfiguration criterion/method	Библиотеки с динамически рекофигурируемыми ячейками/ Libraries with dynamically reconfigurable cells	eFPGA	Схемы, обеспечивающие включение/отключение компонентов/ The circuits that provide for component switching on/off	Реконфигурируемые автоматы + DataPath/ Reconfigurable slot machines + DataPath	Ядра процессоров/ Processor cores	Ядра микроконтроллеров/ Microcontroller cores
Отсутствие специфических требований к технологии/ No specific technology requirements	–	–	+	+	+	+
Отсутствие специфических требований к САПР/ No specific CAD requirements	–	+	+	+	+	+
Накладные расходы по площади и энергопотреблению/ Overhead costs for area size and power consumption	+/–	–	+/–	+	–	–
Возможность обработки потока данных в реальном времени при использовании рабочей частоты равной скорости поступления данных/ Possibility to process a data flow in the real time when using the working frequency equal to a speed of data inflow	+	+	+	+	–	+
Возможность параллельной обработки нескольких потоков событий в реальном времени при использовании рабочей частоты равной скорости поступления данных/ Possibility of parallel processing of multiple flows of events in the real time using the operating frequency equal to a speed of data inflow	+	+	+	+	–	–
Возможность реализации новых режимов/ Possibility to implementation of new modes	+	+	–	+	+	+

него интерфейса. Одному режиму работы блока арбитража соответствует одна конфигурация. При смене режима должна быть записана новая конфигурация.

Блок управления (реконфигурируемый автомат) в рамках предлагаемого подхода используется для выбора номера вектора конфигурации, который должен быть подан на DataPath и блок внутреннего интерфейса на текущем такте. Реконфигурируемый DataPath выполняет обработку входных данных, генерацию значений, поступающих на выходной функциональный интерфейс и на входы блока управления в соответствии с текущими вектором конфигурации, значениями регистров и флагов.

Рассмотрим разработку реконфигурируемого автомата в рамках предлагаемого подхода. Основным ограничением при проектировании автомата является площадь, которая, как было показано выше, преимущественно зависит от разрядности вектора состояния и вектора входных сигналов. Как можно видеть из раздела, в котором описываются архитектура и структура блока арбитража, количество состояний автомата в различных режимах – от трех до пяти. Таким образом, разрядность вектора состояния

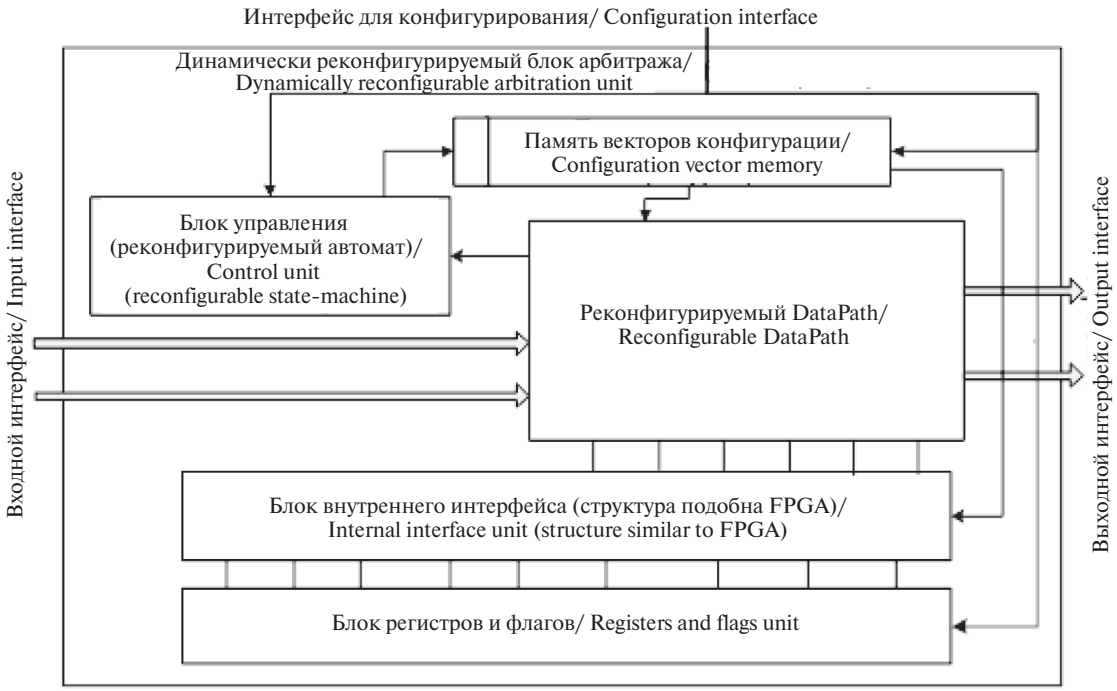

Рисунок 3. Обобщенная схема блока арбитража
Figure 3. Generalized chart of arbitration unit

не превосходит трех. Количество входных сигналов, на которые необходима реакция блока управления, как правило, довольно велико. Оно зависит от количества сигналов запросов ресурса, дополнительных событий, таких как истечение таймеров, которые необходимо контролировать в различных режимах. Условия перехода из состояния в состояние могут быть комплексными (представлять собой некоторую логическую функцию над несколькими сигналами).

Для того чтобы сократить количество входных сигналов, мы предлагаем генерировать входные сигналы, соответствующие значениям логических функций, определяющих переходы между состояниями для блока управления в DataPath. В этом случае разрядность вектора входных сигналов равна максимальному количеству переходов из одного состояния минус один.

Проведенный нами анализ возможных алгоритмов арбитража показал, что разрядность вектора входных сигналов не превосходит двух-трех. Таким образом, количество строк в таблице соответствия, необходимой для реализации автомата управления, не превосходит 32–64. Накладные расходы на ее реализацию приемлемы для большинства систем.

Рассмотрим организацию DataPath в рамках предлагаемого подхода. В DataPath реализуются функции:

- выбора запроса с учетом приоритетов и дополнительных параметров;
- определения текущих значений приоритетов;
- подсчета дополнительных параметров (например, количества переданных слов данных, количества тактов, в течение которых использовался ресурс и др.);
- обработки входных сигналов для формирования вектора входных сигналов для блока управления;
- формирования выходных сигналов.

Реконфигурируемый DataPath состоит из ФМ, которые также могут быть реконфигурируемыми для реализации различных функций и реконфигурируемой структуры межсоединений. В рамках предлагаемого подхода для организации межсоединений между ФМ внутри DataPath и для соединений ФМ с блоком регистров и флагов используется единая реконфигурируемая структура межсоединений, реализованная в блоке внутреннего интерфейса.

Для того чтобы определить набор и функциональность ФМ, входящих в состав DataPath, разработчику необходимо выполнить следующие действия:

- проанализировать существующие алгоритмы арбитража, которые должны быть реализованы, и выполнить прогнозные оценки новых алгоритмов, которые могут появиться в ходе эксплуатации разрабатываемого проекта;
- на основе анализа определить множество функций, которые требуются для реализации алгоритмов;
- определить подмножества функций, которые в соответствии с алгоритмами должны выполняться параллельно (каждая функция в пределах такого

подмножества должна быть реализована в отдельном ФМ);
- в соответствии с определенными подмножествами и особенностями реализации функций определить наборы функций, которые должны быть реализованы в каждом ФМ – определить множество ФМ и набор функций для каждого из них.

Все функции, принадлежащие одному подмножеству, должны быть реализованы в разных ФМ, минимальная мощность множества ФМ, соответственно, равна максимальной мощности подмножества функций.

Для проанализированных нами алгоритмов арбитража [22–28] типовой набор функций включает в себя сравнения (=, <, >), выбор минимального/максимального значения, логические функции, функции счета событий, арифметические функции (+, –, ·, /). Арифметические функции, как правило, используются в алгоритмах, в которых анализируется пропускная способность.

Для реализации логических функций – формирования значений составных условий – мы предлагаем в DataPath использовать функциональный блок со структурой ПЛМ, так как он позволяет реализовывать любые логические функции. Такой блок может использоваться для выбора номера источника запроса, которому будет предоставлен ресурс, а также для формирования значений входных сигналов для блока управления.

Блок внутреннего интерфейса используется для обеспечения связей между регистрами, флагами и функциональными модулями DataPath. Блок состоит из двух частей:

- структуры, связывающей флаги и интерфейсы ФМ, которые имеют разрядность 1 бит;
- структуры, связывающей регистры и интерфейсы ФМ, которые имеют разрядность, соответствующую разрядности слова.

Триггеры, используемые для хранения флагов, подключаются только к структуре, разрядность каналов которой равна 1 бит. Регистры подключаются только к структуре, разрядность каналов которой равна слову. Функциональный модуль своими разными портами может подключаться к обеим структурам в соответствии с их разрядностью. Например, ФМ, выполняющий операции сравнения, своими входами подключается к структуре с разрядностью каналов, равной одному слову, а своим выходом (на который выдается результат сравнения) – к структуре, имеющей разрядность 1 бит.

Обе структуры имеют одинаковую логическую реализацию. В каждой из них имеется несколько логических каналов, количество которых равно количеству входных портов регистров, флагов, ФМ, подключенных к этой структуре. Логический канал обеспечивает возможность в один момент времени соединить соответствующий входной порт с одним любым выходным портом, подключенным к данной структуре. При этом один выходной порт может быть соединен с одним или несколькими входными. На функциональном уровне логический канал представляет собой блок мультиплексирования, на управляющие входы которого подается подвектор вектора конфигурации. При физической реализации блок мультиплексирования может быть реализован как совокупность различных ячеек технологической библиотеки (зависит от используемой технологической библиотеки и САПР).

Такая структура обеспечивает возможность на одном (на каждом) такте записывать новые значения в несколько регистров (во все регистры) и в несколько флагов (во все флаги). На любой вход ФМ может поступать значение из любого регистра, флага или ФМ (в соответствии с разрядностью входа). На любой регистр, флаг может быть записано значение с выхода любого ФМ. Поскольку все регистры и флаги в этой структуре являются симметричными по своим функциональным возможностями, переписывание значений между регистрами и между флагами не имеет смысла, оно не поддерживается в этой структуре. Такая организация структуры связей обеспечивает возможность параллельной обработки нескольких потоков событий (количество параллельно обрабатываемых потоков событий ограничено только количеством ФМ, но не возможностями блока внутреннего интерфейса).

В общем случае реализация таких структур может быть связана с очень большими накладными расходами по площади. Однако для реализации большинства проанализированных нами алгоритмов требуется небольшое количество регистров (менее 20) и небольшое количество ФМ (менее 10). Для такого количества компонентов аппаратные затраты на реализацию блока внутреннего интерфейса не превосходят 10% аппаратных затрат на реализацию блока арбитража в целом, что приемлемо.

При использовании динамически реконфигурируемых блоков спроектированных в соответствии с предлагаемым подходом для них может быть выполнена конфигурация для любого алгоритма арбитража, удовлетворяющего следующим ограничениям:

- набор функциональных модулей, входящих в состав DataPath, позволяет выполнить все функции, необходимые для реализации алгоритма;
- количество векторов конфигурации, необходимое для реализации алгоритма, меньше или

равно количеству слов памяти для хранения векторов конфигурации;
- количество параметров, необходимое для реализации алгоритма, не превосходит количество регистров.

Как было отмечено, используемые алгоритмы арбитража, наборы входных и выходных данных для этих алгоритмов в общем случае не зависят от используемого стандарта внутрикристальных коммуникаций, от того, является ли система коммуникаций шиной, коммутатором или сетью-на-кристалле. Соответственно, предлагаемый подход к проектированию, структура и архитектура блока арбитража могут быть использованы для различных систем коммуникаций.

Оценим накладные расходы на реализацию предложенного подхода и сравним его с реализацией схемы, представленной на рис. 2 (не обеспечивающей возможность поддержки новых режимов). В схеме, представленной на рис. 2, к накладным расходам относятся блоки мультиплексирования. Их площадь мала (менее 10% площади блока арбитража со статическими приоритетами). В схеме, представленной на рис. 3, к накладным расходам относятся блоки памяти для хранения таблицы соответствия, векторов конфигурации и блок внутреннего интерфейса. Необходимо отметить, что блоки, входящие в состав DataPath, используются для реализации различных алгоритмов. Вследствие этого площадь блока DataPath получается существенно меньшей, чем суммарная площадь арбитров, соответствующих разным режимам в схеме, представленной на рис. 2.

Мы выполнили сравнение реализаций для следующих режимов арбитража:

1) статические приоритеты (без прерывания передачи пакетов);
2) статические приоритеты (с прерыванием передачи пакетов);
3) динамические циклические приоритеты (без прерывания передачи пакетов);
4) динамические циклические приоритеты (с прерыванием передачи пакетов);
5) динамические циклические приоритеты с поддержкой механизма планирования (передача данных в соответствии с таблицей таймслотов с прерыванием передачи пакетов);
6) динамические циклические приоритеты с контролем используемой пропускной способности (без прерывания передачи пакетов).

Для данного сравнения выбраны типовые режимы арбитража, используемые в маршрутизаторах СенК для встраиваемых систем. Аналогичные режимы (как правило, за исключением режимов с использованием механизмов планирования) используются в коммутаторах для встраиваемых систем на базе стандартов AXI, AHB, WISHBONE [22–28].

В данном списке режимы арбитража расположены в порядке увеличения площади, если арбитр реализуется как отдельный блок. Как можно видеть, часть режимов обладает повторяющимися требованиями (например, необходимостью поддержки прерывания передачи). Отметим, что данный набор режимов является довольно распространенным, отдельные требования могут быть характерными для различных режимов [1, 3–5]. Сравнение варианта реализации с мультиплексированием режимов (см. рис. 2) и предлагаемого (см. рис. 3) выполнялось для реализации с использованием различных технологических библиотек (180–45 нм), соотношение результатов, полученных для различных библиотек, различается незначительно (в пределах 5%). Полученные результаты представлены на рис. 4, на котором указана относительная площадь полученных блоков, приведенная к условным единицам. (Площадь блока 1 равна пяти условным единицам.)

Как можно видеть из этих графиков, если необходима поддержка режимов 1, 2, 3, площадь предлагаемого варианта практически в два раза превосходит площадь варианта с мультиплексированием. Однако с увеличением количества режимов площадь предлагаемого варианта растет заметно медленнее, чем площадь варианта с мультиплексированием. Это происходит за счет использования функциональных модулей, входящих в состав DataPath и регистров при реализации различных режимов. В результате, если необходимо поддерживать все рассмотренные шесть режимов, то площадь предлагаемого варианта в 1,46 раза меньше площади варианта с мультиплексированием. Кроме того, если в дальнейшем в ходе функционирования системы понадобится добавить новые режимы арбитража, например следующие:

7) динамические циклические приоритеты с поддержкой механизма планирования (передача данных в соответствии с таблицей таймслотов, без прерывания передачи пакетов);
8) динамические циклические приоритеты с контролем используемой пропускной способности (с прерыванием передачи пакетов), при использовании варианта с мультиплексированием это невозможно. Если же используется предлагаемый подход, это можно сделать в реализации, поддерживающей варианты 1–6.

Таким образом, предлагаемый подход целесообразно использовать, если необходима поддержка

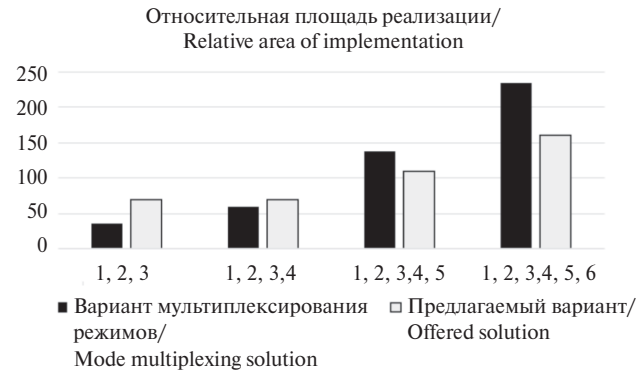

Рисунок 4. Площади блоков арбитража при использовании варианта с мультиплексированием режимов и предложенного варианта
Figure 4. Areas of arbitration units when using the solution with mode multiplexing and the proposed solution

довольно большого числа режимов и если в ходе функционирования системы могут появляться дополнительные режимы.

Выводы

В статье рассмотрены требования к блокам арбитража для встраиваемых систем. Показано, что для многих систем, разрабатываемых с использованием технологии ASIC, необходима динамическая реконфигурация блоков арбитража. Сформулированы критерии, которым должны удовлетворять подходы к динамической реконфигурации для блоков арбитража, рассмотрены существующие подходы к проектированию динамически реконфигурируемых блоков при использовании технологии ASIC. С учетом сформулированных критериев предложен подход, основанный на использовании динамически реконфигурируемых автоматов и DataPath. Данный подход может быть использован для реализации блоков арбитража для коммуникационных систем на основе различных стандартов. Выполнена оценка накладных расходов на реализацию блока по сравнению с использованием подхода, при котором каждому режиму ставится в соответствие отдельный блок, его реализующий. Определены условия, при которых целесообразно использование предложенного подхода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Azarian A.*, *Ahmadi M.* Reconfigurable Computing Architecture Survey and introduction. Publication 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 269–274. DOI: 10.1109/ICCSIT.2009.5234721.
2. *Stensgaard M. B.* ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology. Publication Second ACM/IEEE International Symposium on Networks-on-Chip, 2008, pp. 55-64. DOI: 10.1109/NOCS.2008.4492725.
3. *Cota E.*, *De Morais Amory A.*, *Lubaszewski M. S.* Reliability, Availability and Serviceability of Networks-on-Chip. Springer, 2012, 209 p. DOI: 10.1007/978-1-4614-0791-1.
4. *Jafri S.*, *Liang Guang L.*, *Hemani A.*, *Paul K.*, *Plosila J.*, *Tenhunen H.* Energy-aware fault-tolerant network-on-chips for addressing multiple traffic classes. Microprocessors and Microsystems, 2013, vol. 37, iss. 8, pp. 811–822.
5. *Yoonjin K.* Reconfigurable Multi-Array Architecture for Low-Power and High-Speed Embedded Systems. Journal of semiconductor technology and science, 2011, vol. 11, no. 3, pp. 207–220.
6. *O'Connor I.*, *Hassoune I.*, *Navarro D.* Fine-Grain Reconfigurable Logic Cells Based on Double-Gate MOSFETs. VLSI-SoC: Design Methodologies for SoC and SiP. Springer, Berlin, Heidelberg, 2010, vol. 313, pp. 97–113.
7. *Hassoune I.*, *O'Connor I.* Double-Gate MOSFET Based Reconfigurable Cells. Electronics Letters, 2007, no. 43 (23), pp. 1273–1274.
8. *Speedcore* eFPGA Datasheet (DS003). Achronix Semiconductor Corporation [Электронный ресурс]. URL: https://www.achronix.com/doc/speedcore-efpga-datasheet-ds003/ (дата обращения: 08.07.2019).
9. *Electronic* Lab. open source hardware projects [Электронный ресурс]. URL: http://www.electronics-lab.com/taking-advantage-embedded-fpga-efpga (дата обращения: 08.07.2019).
10. *ARC Processor* Core. Fujitsu Microelectronics America, INC., 2016 [Электронный ресурс]. URL: https://www.fujitsu.com/cn/en/Images/arc_rev3-en.pdf (дата обращения: 11.07.2019).
11. *Sklyarov V.*, *Skliarova I.* Synthesis of parallel hierarchical finite state machines. In Proceedings of the 2013 21st Iranian Conference on Electrical Engineering, ICEE 2013, pp. 1–8. DOI: 10.1109/IranianCEE.2013.6599683.
12. *Glaser J.*, *Damm M.*, *Haase J.*, *Grimm C.* TR-FSM: Transition-based Reconfigurable finite state machine. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2011, vol. 4, no. 3 pp. 1–15. DOI: 10.1145/2000832.2000835.
13. *Design* of Reconfigurable Logic Controllers. In: Karatkevich A., Bukowiec A., Doligalski M., Tkacz J., ed. Berlin, Springer International Publishing AG, 2016, 185 p.
14. *Garcia-Vargas I.*, *Senhadji-Navarro R.* Finite state machines with input multiplexing: A performance Study. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, vol. 34, no. 5, pp. 867–871.
15. *Gupta S.*, *Pareek V.*, *Jain S. C.*, *Jain D.* Realization of sequential reversible circuit from finite state machine. In Proceedings of the International Computer Science and Engineering Conference, ICSEC, 2014, Khon Kaen, Thailand, pp. 458–463. DOI: 10.1109/ICSEC.2014.7024295.
16. *Salauyou V.* Synthesis of high-speed finite state machines in FPGAs by state splitting. In Computer Information Systems and IndustrialManagement: 15th IFIPTC8 International Conference, CISIM, 2016, pp. 741–751. DOI: 10.1007/978-3-319-45378-1_64.
17. *Xydis S.*, *Economakos G.*, *Soudris D.*, *Pekmestzi K.* High Performance and area efficient flexible DSP data path synthesis. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, vol. 19, no. 3, pp. 429–442.

18. *Leibson S.*, *Kim J.* A New Era in chip Design. IEEE, 2005, pp. 51–59.
19. Xtensa LX7 Processor [Электронный ресурс]. URL: https://ip.cadence.com/uploads/1099/TIP_PB_Xtensa_lx7_FINAL-pdf (дата обращения: 11.07.2019).
20. ARC Processor Core. Fujitsu Microelectronics America, INC., 2016 [Электронный ресурс]. URL: https://www.fujitsu.com/cn/en/Images/arc_rev3-en.pdf (дата обращения: 11.07.2019).
21. *Kapasi U. J.*, *Rixner S.*, *Dally W. J.*, *Khailany B.*, *Jung Ho Ahn, Mattson P.*, *Owens J. D.* Programmable stream processors. Computer, 2003, vol. 36, iss. 8, pp. 54–62. DOI: 10.1109/MC.2003.1220582.
22. *Pasricha S.*, *Dutt N.* On-Chip Communication Architectures. System-on-Chip Interconnect. Elsevier, 2008, 544 p.
23. *Wiefereink A.*, *Meyr H.*, *Leupers R.* Retargetable Processor System Integration into Multi-Processor System-on-Chip Network, Springer Netherlands, 2008. 162 p.
24. *Rios-Navarro A.*, *Tapiador-Morales R.*, *Jimenez-Fernandez A.*, *Dominguez-Morales M.*, *Amaya C.*, *Linares-Barranco A*. Performance evaluation over HW/SW co-design SoC memory transfers for a CNN accelerator. IEEE 18th International Conference on Nanotechnology (IEEE-NANO), 2018. DOI: 10.1109/NANO.2018.8626313.
25. *Rohita P. Patil*, *Pratima V. Sangamkar*. A Review of System-On-Chip Bus Protocols. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 4, iss. 1, Jan 2015, pp. 271–281.
26. *Chia-Hsin Owen Chen, Sunghyun Park, Tushar Krishna, Suvinay Subramanian, Anantha P. Chandrakasan, Li-Shiuan Peh*. SMART: A Single-Cycle Reconfigurable NoC for SoC Applications. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013, 7 p. DOI: 10.7873/DATE.2013.080.
27. *Vestias M. P.*, *Neto H. C.* A Generic Network-on-Chip Architecture for Reconfigurable Systems: Implementation and Evaluation. 2006 International Conference on Field Programmable Logic and Applications, 2006, pp. 1–4. DOI: 10.1109/FPL.2006.311303.
28. *Hui Liu*, *Linquan Xie*, *Jiansheng Liu*, *Lei Ding.* Application of Butterfly Clos-Network in Network-on-Chip. ScientificWorld Journal, 2014, vol. 2014, pp. 1–11.

REFERENCES

1. Azarian A., Ahmadi M. Reconfigurable Computing Architecture Survey and introduction. *Publication 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 269–274. DOI: 10.1109/ICCSIT.2009.5234721.
2. Stensgaard M. B. ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology. *Publication Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008, pp. 55-64. DOI: 10.1109/NOCS.2008.4492725.
3. Cota E., De Morais Amory A., Lubaszewski M. S. Reliability, Availability and Serviceability of Networks-on-Chip. Springer, 2012, 209 p. DOI: 10.1007/978-1-4614-0791-1.
4. Jafri S., Liang Guang L., Hemani A., Paul K., Plosila J., Tenhunen H. Energy-aware fault-tolerant network-on-chips for addressing multiple traffic classes. *Microprocessors and Microsystems*, 2013, vol. 37, iss. 8, pp. 811–822.
5. Yoonjin K. Reconfigurable Multi-Array Architecture for Low-Power and High-Speed Embedded Systems. *Journal of semiconductor technology and science*, 2011, vol. 11, no.3, pp. 207–220.
6. O'Connor I., Hassoune I., Navarro D. Fine-Grain Reconfigurable Logic Cells Based on Double-Gate MOSFETs. *VLSI-SoC: Design Methodologies for SoC and SiP*. Springer, Berlin, Heidelberg, 2010, vol. 313, pp. 97–113.
7. Hassoune I., O'Connor, I. Double-Gate MOSFET Based Reconfigurable Cells. *Electronics Letters*, 2007, no. 43 (23), pp. 1273–1274.
8. Speedcore eFPGA Datasheet (DS003). Achronix Semiconductor Corporation. Available at: https://www.achronix.com/doc/speedcore-efpga-datasheet-ds003/ (accessed 08.07.2019).
9. Electronic Lab. open source hardware projects. Available at: http://www.electronics-lab.com/taking-advantage-embedded-fpga-efpga (accessed 08.07.2019).
10. ARC Processor Core. Fujitsu Microelectronics America, INC., 2016. Available at: https://www.fujitsu.com/cn/en/Images/arc_rev3-en.pdf (accessed 11.07.2019).
11. Sklyarov V., Skliarova I. Synthesis of parallel hierarchical finite state machines. In *Proceedings of the 2013 21st Iranian Conference on Electrical Engineering*, ICEE 2013, pp. 1–8. DOI: 10.1109/IranianCEE.2013.6599683.
12. Glaser J., Damm M., Haase J., Grimm C. TR-FSM: Transition-based Reconfigurable finite state machine. *ACM Transactions on Reconfigurable Technology and Systems* (*TRETS*), 2011, vol. 4, no. 3. pp. 1–15. DOI: 10.1145/2000832.2000835.
13. Design of Reconfigurable Logic Controllers. In: Karatkevich A., Bukowiec A., Doligalski M., Tkacz J., ed. Springer International Publishing AG, 2016, 185 p.
14. Garcia-Vargas I., Senhadji-Navarro R. Finite state machines with input multiplexing: A performance Study. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015, vol. 34, no. 5, pp. 867–871.
15. Gupta S., Pareek V., Jain S. C., Jain D. Realization of sequential reversible circuit from finite state machine. *In Proceedings of the International Computer Science and Engineering Conference*, ICSEC, 2014, Khon Kaen, Thailand, pp. 458–463. DOI: 10.1109/ICSEC.2014.7024295.
16. *Salauyou V.* Synthesis of high-speed finite state machines in FPGAs by state splitting. *In Computer Information Systems and IndustrialManagement: 15th IFIPTC8 International Conference*, CISIM, 2016, pp. 741–751. DOI: 10.1007/978-3-319-45378-1_64.
17. Xydis S., Economakos G., Soudris D., Pekmestzi K. High Performance and area efficient flexible DSP data path synthesis. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2011, vol. 19, no. 3, pp. 429–442.
18. *Leibson S.*, *Kim J.* A New Era in chip Design. IEEE, 2005, pp. 51–59.
19. Xtensa LX7 Processor. Available at: https://ip.cadence.com/uploads/1099/TIP_PB_Xtensa_lx7_FINAL-pdf / (accessed: 11.07.2019).
20. ARC Processor Core. Fujitsu Microelectronics America, INC., 2016. Available at: https://www.fujitsu.com/cn/en/Images/arc_rev3-en.pdf (accessed: 11.07.2019).
21. Kapasi U. J., Rixner S., Dally W. J., Khailany B., Jung Ho Ahn, Mattson P., Owens J. D. Programmable stream processors. *Computer*, 2003, vol. 36, iss. 8, pp. 54–62. DOI: 10.1109/MC.2003.1220582.

22. Pasricha S., Dutt N. On-Chip Communication Architectures. System-on-Chip Interconnect. Elsevier, 2008, 544 p.
23. Wiefereink A., Meyr H., Leupers R. Retargetable Processor System Integration into Multi-Processor System-on-Chip Network. Springer Netherlands, 2008. 162 p.
24. Rios-Navarro A., Tapiador-Morales R., Jimenez-Fernandez A., Dominguez-Morales M., Amaya C., Linares-Barranco A. Performance evaluation over HW/SW co-design SoC memory transfers for a CNN accelerator. *IEEE 18th International Conference on Nanotechnology* (*IEEE-NANO*), 2018. DOI: 10.1109/NANO.2018.8626313.
25. Rohita P. Patil, Pratima V. Sangamkar. A Review of System-On-Chip Bus Protocols. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2015, vol. 4, iss. 1, pp. 271–281.
26. Chia-Hsin Owen Chen, Sunghyun Park, Tushar Krishna, Suvinay Subramanian, Anantha P. Chandrakasan, Li-Shiuan Peh. SMART: A Single-Cycle Reconfigurable NoC for SoC Applications. *Design, Automation & Test in Europe Conference & Exhibition* (*DATE*), 2013, 7 p. DOI: 10.7873/DATE.2013.080.
27. Vestias M. P., Neto H. C. A Generic Network-on-Chip Architecture for Reconfigurable Systems: Implementation and Evaluation. *2006 International Conference on Field Programmable Logic and Applications*, 2006, pp. 1–4. DOI: 10.1109/FPL.2006.311303.
28. Hui Liu, Linquan Xie, Jiansheng Liu, Lei Ding. Application of Butterfly Clos-Network in Network-on-Chip. *ScientificWorld Journal*, 2014, vol. 2014, pp. 1–11.

ИНФОРМАЦИЯ ОБ АВТОРЕ

Суворова Елена Александровна, к.т.н., доцент, Санкт-Петербургский государственный университет аэрокосмического приборостроения, 190000, Санкт-Петербург, ул. Большая Морская, д. 67, тел.: +7 (911) 986-66-67, e-mail: elena.suvorova@guap.ru.

AUTHOR

Elena A. Suvorova, Ph.D. (Engineering), Assistant professor, Saint-Petersburg State University of Aerospace Instrumentation, 67, ulitsa Bolshaya Morskaya, St. Petersburg, 190000, Russia, tel.: +7 (911) 986-66-67, e-mail: elena.suvorova@guap.ru.

Поступила 18.03.2019; принята к публикации 28.06.2019; опубликована онлайн 21.08.2019.
Submitted 18.03.2019; revised 28.06.2019; published online 21.08.2019.